**ONLINE BANKING MANAGEMENT SYSTEM**

**A MINI PROJECT REPORT**

***Submitted by***

**HARINI P (953622104029)**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

****

**RAMCO INSTITUTE OF TECHNOLOGY,**

**RAJAPALAYAM**

**DECEMBER 2023**

**CS3381 OBJECT ORIENTED PROGRAMMING LABORATORY**

# BONAFIDE CERTIFICATE

Certified that this project report titled **“Online Banking Management System”** is the bonafide work of **Harini P (953622104029),** who carried out the project work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE**  **Dr.K.Vijayalakshmi, M.E, Ph.D.**  **HEAD OF THE DEPARTMENT**  Professor & Head  Department of Computer Science and Engineering  Ramco Institute of Technology North Venganallur Village  Rajapalayam-626117. | **SIGNATURE**  **Mrs.B.Vijayalakshmi, M.E. SUPERVISOR**  Assistant Professor (SG)  Department of Computer Science and Engineering  Ramco Institute of Technology North Venganallur Village  Rajapalayam-626117. |

**ABSTRACT**

The Online Banking System is a comprehensive solution that addresses the evolving needs of modern banking by providing users with a secure and efficient platform for managing their financial transactions. Leveraging Java and MySQL technologies, the system offers features such as account creation, secure login/logout mechanisms, profile management, transactional capabilities, withdrawal and deposit options and also providing help option to guide the user. Login/Logout and all these actions are perfectly done using the database integration. Login details and the transaction details has been stored in database for effective usage.

**1.INTRODUCTION:**

In an age characterized by digital transformation, the Online Banking System emerges as a transformative force, reshaping the dynamics of financial interactions. This innovative project introduces a sophisticated Online Banking System, meticulously crafted to streamline processes and enhance user experiences. The synergy of Java and MySQL technologies empowers the system to offer a diverse range of functionalities, demonstrating the profound implications on contemporary banking practices.

**2.PROJECT DESCRIPTION:**

The primary objectives of the Online Banking System include streamlining banking procedures, enhancing user convenience, and facilitating efficient money management. By providing features such as account creation, secure login/logout mechanisms, profile management, and transactional capabilities, the project aims to create a seamless and responsive banking experience.

**3.MODULES:**

**Home Page**

The central hub providing users with an overview of available features, notifications, and access points to different modules.

**Account Management and Onboarding**

Creating a new account is a user-friendly process, guided by an intuitive interface. The system captures essential user details, assigns unique account numbers, and ensures a smooth onboarding experience.

**User Authentication and Security**

The system prioritizes user security with robust login/logout functionalities. Secure authentication measures, including password hashing and protection against common vulnerabilities, ensure the confidentiality and integrity of user accounts.

**Profile Management**

Enables users to update and manage their profiles, incorporating functionalities for password resets and updates.

**Balance Inquiry**

Provides real-time access to account balances, empowering users to make informed financial decisions.

**Deposit Transactions**

Streamlines the process of depositing funds into user accounts securely and efficiently.

**Withdrawal Mechanism**

Facilitates secure and swift withdrawals, ensuring users have prompt access to their financial resources.

**Fund Transfer**

Supports seamless fund transfers between accounts, enhancing overall financial agility.

**4.TECHNOLOGY STACK:**

**Programming Language:** **Java**

Serving as the backbone for backend logic, integrating seamlessly with the GUI and ensuring optimal system performance.

**Packages Used:**

**javafx.scene.control:** Utilized for TextField, PasswordField, Button, Label, and other GUI components.

**javafx.scene.layout:** Used for organizing the layout of the graphical user interface.

**javafx.geometry.Insets:** This class is used to represent the insets of a rectangular area. In JavaFX, the Insets class is typically used in layout management to define the spacing around the edges of a node or a region within a layout container. It allows you to specify the top, right, bottom, and left insets in pixels.

**javafx.scene.Cursor:** Enables the customized cursor option.

**javafx.scene.paint.Color:** It provides various ways to define colors, including predefined colors, RGB values, HSB values, and named colors. **javafx.scene.text**: This package in JavaFX provides classes for working with text and fonts in JavaFX applications. Here are some key classes in this package:

**javafx.scene.text.Font:** This class represents a font and is used to define the style, size, and family of a text.

**javafx.stage.Modality:**This enum in JavaFX is used to define the modality of a window or a stage. Modality refers to the behavior of a window concerning user interaction with other windows in the application.

**1.APPLICATION\_MODAL:**

The window is modal to the entire application. It blocks user interaction with any other window in the application, including windows from other stages.

**java.sql:** This package in Java provides the classes and interfaces for the JDBC (Java Database Connectivity) API. JDBC is a standard Java API for interacting with relational databases.

**java.sql.DriverManager**: This class manages a list of database drivers. It is used to establish a connection to a database.

**java.sql.Connection**: This interface represents a connection to a database. You use this interface to execute SQL statements and manage transactions.

**javafx.scene.control.Alert.AlertType**: This enum is used to specify the type or style of an Alert dialog. The Alert class is part of the JavaFX UI controls and is commonly used to show informative, warning, error, or confirmation messages to the user.

**MYSQL DATABASE INTEGRATION**

**Database Connectivity Using JDBC**

**MySQL Connector/J Integration**

The Online Banking System leverages MySQL as its relational database management system. To establish connectivity between the Java application and the MySQL database, the MySQL Connector/J(JAR file), a JDBC driver for MySQL, is utilized. Here are the steps to connect JDBC in Appache NetBeans.

**Step 1: Download and Install MySQL Connector/J**

Download the MySQL Connector/J: mysql-connector-java:8.0.28from the official MySQL website (MySQL Connector/J Download) and ensure it is included in my project's classpath(Adding that JAR file in the Library).

**Step 2: Adding Required Dependencies**

Under Project files, Choose pom.xml file, and add below dependency.

<dependencies>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.28</version>

</dependency>

</dependencies>

This dependency is for MySQL Connector/J, which is a JDBC (Java Database Connectivity) driver for connecting Java applications to MySQL databases. The version 8.0.28 indicates the desired version of the MySQL Connector/J.

**Step 3: Adding Requirement for SQL**

In module-info.java file (Under Source packages-> default package), add requirement for sql - requires java.sql;

**Step 4: Connectivity Check**

Under Dependency folder, Check whether the Connector/J file is added properly(JAR file). Then, check the Services Tab for the database that we are going to work with. It can be found by clicking other connection option.

**Step 5: Queries for Operations**

By using suitable queries, the inserting, updating, validating data can be done.

These are the steps for connecting Our Database with JDBC.

**MySQL Database:** Anchoring the system with a reliable and scalable storage solution, housing user data, account details, and comprehensive transaction records. After installing MySql workbench, Under Schema, the required queries can be written.

**Database Schema**

The MySQL database plays a pivotal role in managing critical information:

**Database creation Query**

* To create a database – ‘CREATE DATABASE login;’
* This will create a new database named ‘login’.

**Table creation under login DB**

* To create tables under login database - use login
* This query is used before creating a table. The tables are created only if they know where to create the tables. This is the purpose of that query.

1. **Logindetails Table Creation**: Repository for user-specific details such as usernames, passwords, email addresses, account number, pin number, balance and contact information. The below query will create the table logindetails with the respective columns.

create table logindetails

(

id INT AUTO\_INCREMENT PRIMARY KEY,

name varchar(50),

email varchar(100),

password varchar(100),

accountNum int,

pin int,

bal int,

mobile varchar(100),

address varchar(100),

timestamp DATETIME )

1. **Transaction Table Creation**: Captures detailed transaction information, from timestamps to transaction types and amounts. The below query will create the table transaction with the respective columns.

create table transaction(

id INT AUTO\_INCREMENT PRIMARY KEY,

accountNum int,

pin int,

ToAccount varchar(50),

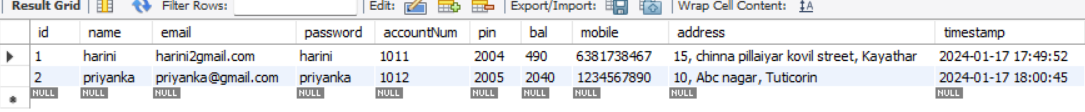
transactiontype varchar(50),

amount int)

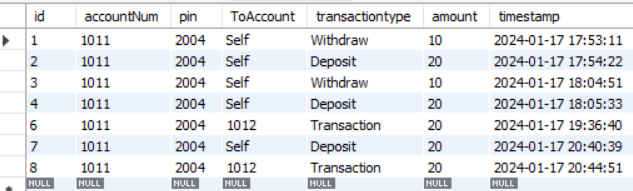
**Viewing Table Content**

To view table content:

* select \* from logindetails



* select \* from transaction

****

**5.CODING**

package com.mycompany.banking;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import javafx.application.Application;

import javafx.application.Platform;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Cursor;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import static javafx.scene.control.Alert.AlertType.INFORMATION;

import javafx.scene.layout.\*;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.Text;

import javafx.stage.Modality;

import javafx.stage.Stage;

import java.sql.\*;

import javafx.scene.control.Alert.AlertType;

public class App extends Application {

int accNo;

int balance;

int user;

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage primaryStage) {

loadAccNoFromFile();

primaryStage.setTitle("Online Banking");

BorderPane root = new BorderPane();

root.setStyle("-fx-background-color: white;");

GridPane loginGrid = createLoginForm(primaryStage);

loginGrid.setStyle("-fx-background-color: white; -fx-border-

color: #cccccc; -fx-border-width: 1; -fx-border-radius: 5; -fx-

padding: 20;");

root.setCenter(loginGrid);

Scene scene = new Scene(root, 400, 300);

primaryStage.setScene(scene);

// Show the stage

primaryStage.show();

}

private void loadAccNoFromFile() {

try (BufferedReader reader = new BufferedReader(new FileReader("C:\\Users\\harin\\OneDrive\\Desktop\\java\\accNo.txt")))

{

String accNoStr = reader.readLine();

if (accNoStr != null && !accNoStr.isEmpty())

{

accNo = Integer.parseInt(accNoStr);

}

} catch (IOException | NumberFormatException e) {

e.printStackTrace();

}

}

private GridPane createLoginForm(Stage primaryStage) {

GridPane grid = new GridPane();

grid.setAlignment(Pos.CENTER);

grid.setHgap(10);

grid.setVgap(10);

grid.setPadding(new Insets(25, 25, 25, 25));

Text scenetitle = new Text("Welcome to Online Banking");

scenetitle.setFont(Font.font("Tahoma", 20));

grid.add(scenetitle, 0, 0, 2, 1);

Label userName = new Label("User Name:");

grid.add(userName, 0, 1);

TextField userTextField = new TextField();

userTextField.setStyle("-fx-background-color: #f8f8f8; -fx-border-

color: #dddddd; -fx-border-width: 1; -fx-border-radius: 3; -

fx-padding: 5;");

grid.add(userTextField, 1, 1);

Label password = new Label("Password:");

grid.add(password, 0, 2);

PasswordField passwordField = new PasswordField();

passwordField.setStyle("-fx-background-color: #f8f8f8; -fx-border-

color: #dddddd; -fx-border-width: 1; -fx-border-radius: 3; -fx-padding: 5;");

grid.add(passwordField, 1, 2);

Button signInButton = new Button("Login");

Button createAccountButton = new Button("Create Account");

signInButton.setStyle("-fx-background-color:slateblue;-fx-text-fill:white");

signInButton.setCursor(Cursor.HAND);

createAccountButton.setCursor(Cursor.HAND);

createAccountButton.setStyle("-fx-background-color:slateblue;-fx-

text-fill:white");

HBox hbBtn = new HBox(10);

hbBtn.setAlignment(Pos.BOTTOM\_RIGHT);

hbBtn.getChildren().addAll(signInButton, createAccountButton);

grid.add(hbBtn, 1, 4);

signInButton.setOnAction(e -> handleLogin(userTextField, passwordField,primaryStage));

createAccountButton.setOnAction(e -> showCreateAccountDialog(primaryStage));

return grid;

}

private void handleLogin(TextField username, PasswordField password,Stage stage) {

try{

String us=username.getText();

String pass=password.getText();

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con=DriverManager.getConnection

("jdbc:mysql://localhost:3306/login?useSSL=false","root",

"r@m4112004");

Statement stm=con.createStatement();

String sql="select \* from logindetails where

name='"+us+"' and password='"+pass+"'";

ResultSet rs=stm .executeQuery(sql)

if(rs.next())

{

user=rs.getInt("accountNum");

stage.close();

showMainDashboard();

}

else

{

if(us.isEmpty()|| pass.isEmpty())

{

displayErrorAlert("Please fill all the fields");

}

else{Alert alert = new Alert(AlertType.ERROR);

alert.setTitle("Invalid Credentials");

alert.setHeaderText(null);

alert.setContentText("Please check your

username and password.");

alert.showAndWait();

username.setText("");

password.setText("");}

}

}

catch(Exception e){

System.out.println(e.getMessage());

}

}

private void showMainDashboard() {

BorderPane mainDashboard = new BorderPane();

// Create MenuBar

MenuBar menuBar = createMenuBar();

mainDashboard.setTop(menuBar);

// Set the center of the main dashboard (you can customize this part)

mainDashboard.setCenter(new Label("Welcome to the Online Banking"));

// Create GridPane for the center content

GridPane centerGrid = new GridPane();

centerGrid.setAlignment(Pos.BOTTOM\_RIGHT);

centerGrid.setHgap(10);

centerGrid.setVgap(10);

centerGrid.setPadding(new Insets(25, 25, 25, 25));

// Add Logout button

Button logout = new Button("Logout");

logout.setStyle("-fx-background-color:slateblue;-fx-text-fill:white");

logout.setCursor(Cursor.HAND);

// Add event handler for Logout button

logout.setOnAction(e -> handleLogout(mainDashboard));

// Add the Logout button to the GridPane

centerGrid.add(logout, 0, 0);

// Set the center of the main dashboard to the GridPane

mainDashboard.setBottom(centerGrid);

// Create scene and set it on the stage

Scene mainScene = new Scene(mainDashboard, 600, 400);

Stage mainStage = new Stage();

mainStage.setScene(mainScene);

mainStage.setTitle("Online Banking");

// Close the login stage and show the main dashboard stage

Stage loginStage = (Stage) menuBar.getScene().getWindow();

loginStage.close();

mainStage.show();

}

private void handleLogout(BorderPane mainDashboard) {

// Close the main stage

Stage mainStage = (Stage) mainDashboard.getScene().getWindow();

mainStage.close();

// Show the login form again

start(new Stage());

}

private MenuBar createMenuBar() {

MenuBar menuBar = new MenuBar();

Menu homeMenu = createMenu("Home");

Menu withdrawMenu = createMenu("Withdraw");

Menu depositMenu = createMenu("Deposit");

Menu balanceMenu = createMenu("Balance");

Menu transactionMenu = createMenu("Transaction");

Menu helpMenu = createMenu("Help");

menuBar.getMenus().addAll(homeMenu, withdrawMenu,

depositMenu, balanceMenu, transactionMenu, helpMenu);

return menuBar;

}

private Menu createMenu(String menuName) {

Menu menu = new Menu(menuName);

menu.setStyle("-fx-font-size: 14; -fx-text-fill: #4caf50;");

MenuItem menuItem = new MenuItem("Go to " + menuName);

menu.getItems().add(menuItem);

menuItem.setOnAction(e -> handleMenuAction(menuName));

return menu;

}

private void handleMenuAction(String menu) {

// Handle menu actions here

if ("Withdraw".equals(menu)) {

handleWithdrawal();

}

if ("Deposit".equals(menu)) {

handleDeposit();

}

if("Balance".equals(menu))

{

handleBalance();

}

if("Transaction".equals(menu))

{

handleTransaction();

}

if("Help".equals(menu))

{

handleHelp();

}

}

private void handleHelp(){

Stage helpStage=new Stage();

GridPane helpGrid=new GridPane();

helpGrid.setAlignment(Pos.CENTER);

helpGrid.setHgap(10);

helpGrid.setVgap(10);

helpGrid.setPadding(new Insets(25, 25, 25, 25));

Text sceneTitle = new Text("Guidence");

sceneTitle.setFont(Font.font("Tahoma", 20));

helpGrid.add(sceneTitle, 0, 0, 1, 1);

String userGuideContent =

"Welcome to the Online Banking User Guide!\n\n" +

"1. Logging In:\n" +

" - Enter your username and password.\n" +

" - Click the 'Login' button to access your account.\n\n" +

"2. Main Dashboard:\n" +

" - View your account summary and recent transactions.\n" +

" - Use the menu bar for additional actions.\n\n" +

"3. Transactions:\n" +

" - Make transfers, check balances, and view transaction history.\n\n" +

"4. Logout:\n" +

" - Click the 'Logout' button to securely log out of your account.\n\n" +

"For more detailed instructions, please visit our website or contact support.\n"

+ "Email: support@mybank.com “+”\nPhone: 123-456-7890";

Button CancelButton = new Button("Cancel");

EventHandler<ActionEvent> can=new EventHandler<ActionEvent>(){

@Override

public void handle(ActionEvent t) {

helpStage.close();

}

};

CancelButton.addEventFilter(ActionEvent.ACTION, can);

HBox hbBtn = new HBox(10);

CancelButton.setStyle("-fx-background-color:slateblue;-fx-text-fill:white");

CancelButton.setCursor(Cursor.HAND);

CancelButton.setCursor(Cursor.HAND);

hbBtn.setAlignment(Pos.BOTTOM\_RIGHT);

hbBtn.getChildren().add(CancelButton);

Text text=new Text(userGuideContent);

helpGrid.add(text, 0, 1);

helpGrid.add(hbBtn, 1, 2);

helpGrid.setStyle("-fx-background-color: white; -fx-border-

color: #cccccc; -fx-border-width: 1; -fx-border-radius: 5; -fx-padding: 20;");

Scene helpScene=new Scene(helpGrid,550,500);

helpStage.setTitle("Hellp");

helpStage.setScene(helpScene);

helpStage.show();

}

private void handleTransaction() {

Stage TransStage = new Stage();

TransStage.setTitle("Transaction");

GridPane TransGrid = new GridPane();

TransGrid.setAlignment(Pos.CENTER);

TransGrid.setHgap(10);

TransGrid.setVgap(10);

TransGrid.setPadding(new Insets(25, 25, 25, 25));

Text sceneTitle = new Text("Transaction");

sceneTitle.setFont(Font.font("Tahoma", 20));

TransGrid.add(sceneTitle, 0, 0, 2, 1);

Label pin=new Label("Enter Your Pin:");

PasswordField pinField=new PasswordField();

TransGrid.add(pin, 0, 1);

TransGrid.add(pinField, 1, 1);

Label Toacc = new Label("Account Number(To account):");

TextField accField = new TextField();

TransGrid.add(Toacc, 0, 2);

TransGrid.add(accField, 1, 2);

accField.setPromptText("Acc Number");

Label wl2 = new Label("Amount:");

TransGrid.add(wl2, 0, 3);

TextField amountField = new TextField();

amountField.setPromptText("Transaction Amount");

TransGrid.add(amountField, 1, 3);

Label l1=new Label("Click OK to continue transaction");

TransGrid.add(l1, 0, 5);

Button TransButton = new Button("OK");

HBox hbBtn = new HBox(10);

TransButton.setStyle("-fx-background-color:slateblue;-fx-text-fill:white");

TransButton.setCursor(Cursor.HAND);

Button CancelButton = new Button("Cancel");

EventHandler<ActionEvent> can=new EventHandler<ActionEvent>(){

@Override

public void handle(ActionEvent t) {

TransStage.close();

}

};

CancelButton.addEventFilter(ActionEvent.ACTION, can);

CancelButton.setStyle("-fx-background-color:slateblue;-fx-text-fill:white");

CancelButton.setCursor(Cursor.HAND);

hbBtn.setAlignment(Pos.BOTTOM\_RIGHT);

hbBtn.getChildren().addAll(TransButton,CancelButton);

TransGrid.add(hbBtn, 1, 5);

TransButton.setOnAction(e -> handleTransAction

(TransStage,pinField,accField ,amountField));

TransGrid.setStyle("-fx-background-color: white; -fx-border-color: #cccccc;

-fx-border-width: 1; -fx-border-radius: 5; -fx-padding: 20;");

Scene TransScene = new Scene(TransGrid, 450, 300);

TransStage.setScene(TransScene);

TransStage.initModality(Modality.APPLICATION\_MODAL);

TransStage.showAndWait();

}

private void handleTransAction(Stage TransStage,PasswordField pinField,TextField accField, TextField amountField) {

try {

String pin=pinField.getText();

String acc = accField.getText();

String amountStr = amountField.getText();

if (pin.isEmpty()||acc.isEmpty() || amountStr.isEmpty()) {

displayErrorAlert("Please fill in all fields.");

return;

}

int ac =user;

int pn = Integer.parseInt(acc);

int amt = Integer.parseInt(amountStr);

int pinNo=Integer.parseInt(pin);

Connection con = DriverManager.getConnection

("jdbc:mysql://localhost:3306/login", "root", "r@m4112004");

String checkIfExistsQuery = "SELECT bal FROM logindetails WHERE accountNum= ? AND pin= ?";

try (PreparedStatement checkIfExistsStmt = con.prepareStatement(checkIfExistsQuery)) {

checkIfExistsStmt.setInt(1, ac);

checkIfExistsStmt.setInt(2, pinNo);

ResultSet rs = checkIfExistsStmt.executeQuery();

if (rs.next()) {

int balance = rs.getInt("bal");

if(amt>0) {

String updateQuery = "UPDATE logindetails SET bal = bal - ?

WHERE accountNum= ?";

try (PreparedStatement updateStatement =

con.prepareStatement(updateQuery)) {

updateStatement.setInt(1, amt);

updateStatement.setInt(2, ac);

int rowsAffected = updateStatement.executeUpdate();

if (rowsAffected > 0) {

// Update the balance variable after the withdrawal

balance -= amt;

Alert alert = new Alert(AlertType.INFORMATION);

alert.setTitle("Transaction");

alert.setHeaderText(null);

alert.setContentText("Congratulations! Transaction successful

Your Balance is "+balance);

alert.showAndWait();

TransStage.close();

}

else {

displayErrorAlert("Transaction Failed. Please

check your credentials");

}

}

String updateAcQuery = "UPDATE logindetails SET bal = bal + ? WHERE accountNum= ?";

try (PreparedStatement updateStatement = con.prepareStatement(updateAcQuery)) {

updateStatement.setInt(1, amt);

updateStatement.setInt(2, pn);

int rowsAffected = updateStatement.executeUpdate();

}

String insertQuery = "INSERT INTO transaction(accountNum, pin,Toaccount, transactiontype, amount,timestamp) VALUES (?, ?, ?, ?,?,?)";

try (PreparedStatement insertStmt = con.prepareStatement(insertQuery))

{

insertStmt.setInt(1, ac);

insertStmt.setInt(2, pinNo);

insertStmt.setString(3,acc);

insertStmt.setString(4, "Transaction");

insertStmt.setInt(5, amt);

Timestamp currentTimestamp = new Timestamp

(System.currentTimeMillis());

insertStmt.setTimestamp(6, currentTimestamp);

int rowsAffected =insertStmt.executeUpdate();

}

}

else

{

displayErrorAlert("Please check your balance and transaction amount");

pinField.setText("");

accField.setText("");

amountField.setText("");}

}

else {

displayErrorAlert("Please check your credentials.");

pinField.setText("");

accField.setText("");

amountField.setText("");

}

}

} catch (Exception e) {

e.printStackTrace();

displayErrorAlert("An error occurred. Please try again.");

}

}

private void handleBalance(){

Stage balStage=new Stage();

balStage.setTitle("Balance");

GridPane BalGrid = new GridPane();

BalGrid.setAlignment(Pos.CENTER);

BalGrid.setHgap(10);

BalGrid.setVgap(10);

BalGrid.setPadding(new Insets(25, 25, 25, 25));

Text sceneTitle = new Text("Account Balance");

sceneTitle.setFont(Font.font("Tahoma", 20));

BalGrid.add(sceneTitle, 0, 0, 2, 1);

Button CancelButton = new Button("Cancel");

CancelButton.setStyle("-fx-background-color:slateblue;-fx-text-fill:white");

CancelButton.setCursor(Cursor.HAND);

HBox hbBtn = new HBox(10);

hbBtn.setAlignment(Pos.BOTTOM\_RIGHT);

hbBtn.getChildren().addAll(CancelButton);

BalGrid.add(hbBtn,1,2);

EventHandler<ActionEvent> can=new EventHandler<ActionEvent>(){

@Override

public void handle(ActionEvent t) {

balStage.close();

}

};

CancelButton.addEventFilter(ActionEvent.ACTION, can);

try {

int ac =user;

Connection con = DriverManager.getConnection

("jdbc:mysql://localhost:3306/login", "root", "r@m4112004");

String checkIfExistsQuery = "SELECT bal FROM logindetails WHERE accountNum= ? ";

try (PreparedStatement checkIfExistsStmt = con.prepareStatement

(checkIfExistsQuery)) {

checkIfExistsStmt.setInt(1, ac);

ResultSet rs = checkIfExistsStmt.executeQuery();

if (rs.next()) {

int balance = rs.getInt("bal");

Text b=new Text("Your Balance is "+balance);

b.setFont(Font.font("Tahoma", 15));

BalGrid.add(b, 1, 1, 2, 1);

} else {

displayErrorAlert("Unable to check Balance");

}

}

} catch (Exception e) {

e.printStackTrace();

displayErrorAlert("An error occurred. Please try again.");

}

BalGrid.setStyle("-fx-background-color: white; -fx-border-color: #cccccc;

-fx- border-width: 1; -fx-border-radius: 5; -fx-padding: 20;");

Scene BalScene = new Scene(BalGrid, 300, 200);

balStage.setScene(BalScene);

balStage.initModality(Modality.APPLICATION\_MODAL);

balStage.showAndWait();

}

private void handleDeposit() {

Stage depositStage = new Stage();

depositStage.setTitle("Deposit");

GridPane depositGrid = new GridPane();

depositGrid.setAlignment(Pos.CENTER);

depositGrid.setHgap(10);

depositGrid.setVgap(10);

depositGrid.setPadding(new Insets(25, 25, 25, 25));

Text sceneTitle = new Text("Deposit");

sceneTitle.setFont(Font.font("Tahoma", 20));

depositGrid.add(sceneTitle, 0, 0, 2, 1);

Label pin = new Label("Pin");

PasswordField pinField = new PasswordField();

depositGrid.add(pin, 0, 1);

depositGrid.add(pinField, 1, 1);

pinField.setPromptText("Pin Number");

Label wl2 = new Label("Amount:");

depositGrid.add(wl2, 0, 2);

TextField amountField = new TextField();

amountField.setPromptText("Deposit Amount");

depositGrid.add(amountField, 1, 2);

Button depositButton = new Button("Deposit");

Button CancelButton = new Button("Cancel");

CancelButton.setStyle("-fx-background-color:slateblue;-fx-text-fill:white");

CancelButton.setCursor(Cursor.HAND);

HBox hbBtn = new HBox(10);

depositButton.setStyle("-fx-background-color:slateblue;-fx-text-fill:white");

depositButton.setCursor(Cursor.HAND);

hbBtn.setAlignment(Pos.BOTTOM\_RIGHT);

hbBtn.getChildren().addAll(depositButton,CancelButton);

depositGrid.add(hbBtn, 1, 3);

EventHandler<ActionEvent> can=new EventHandler<ActionEvent>(){

@Override

public void handle(ActionEvent t) {

depositStage.close();

}

};

CancelButton.addEventFilter(ActionEvent.ACTION, can);

depositButton.setOnAction(e -> handleDepositAction(depositStage,pinField, amountField));

depositGrid.setStyle("-fx-background-color: white; -fx-border-color: #cccccc;

-fx-border-width: 1; -fx-border-radius: 5; -fx-padding: 20;");

Scene depositScene = new Scene(depositGrid, 400, 300);

depositStage.setScene(depositScene);

depositStage.initModality(Modality.APPLICATION\_MODAL);

depositStage.showAndWait();

}

private void handleDepositAction(Stage depositStage,PasswordField pinField, TextField amountField) {

try {

String pinNo = pinField.getText();

String amountStr = amountField.getText();

if (pinNo.isEmpty() || amountStr.isEmpty()) {

displayErrorAlert("Please fill in all fields.");

return;

}

int ac =user;

int pn = Integer.parseInt(pinNo);

int amt = Integer.parseInt(amountStr);

Connection con = DriverManager.getConnection

("jdbc:mysql://localhost:3306/login", "root", "r@m4112004");

String checkIfExistsQuery = "SELECT bal FROM logindetails WHERE accountNum= ? AND pin= ?"

try (PreparedStatement checkIfExistsStmt = con.prepareStatement

(checkIfExistsQuery)) {

checkIfExistsStmt.setInt(1, ac);

checkIfExistsStmt.setInt(2, pn);

ResultSet rs = checkIfExistsStmt.executeQuery();

if (rs.next()) {

int balance = rs.getInt("bal");

if(amt>0) {

// Update balance and insert transaction

String updateQuery = "UPDATE logindetails SET bal = bal + ?

WHERE accountNum= ?";

try (PreparedStatement updateStatement = con.prepareStatement

(updateQuery)) {

updateStatement.setInt(1, amt);

updateStatement.setInt(2, ac);

int rowsAffected = updateStatement.executeUpdate();

if (rowsAffected > 0) {

// Update the balance variable after the withdrawal

balance += amt;

Alert alert = new Alert(AlertType.INFORMATION);

alert.setTitle("Withdraw");

alert.setHeaderText(null);

alert.setContentText("Congratulations! Deposit

successful! Your Balance is "+balance);

alert.showAndWait();

depositStage.close();

}

else {

displayErrorAlert("Failed to deposit.");

}

}

String insertQuery = "INSERT INTO transaction(accountNum, pin,Toaccount, transactiontype, amount,timestamp) VALUES (?, ?, ?, ?,?,?)";

try (PreparedStatement insertStmt = con.prepareStatement(insertQuery))

{

insertStmt.setInt(1, ac);

insertStmt.setInt(2, pn);

insertStmt.setString(3,"Self");

insertStmt.setString(4, "Deposit");

insertStmt.setInt(5, amt);

Timestamp currentTimestamp = new Timestamp

(System.currentTimeMillis());

insertStmt.setTimestamp(6, currentTimestamp);

int rowsAffected =insertStmt.executeUpdate();

}

}

} else {

displayErrorAlert("Please check your credentials.");

pinField.setText("");

amountField.setText("");

}

}

} catch (Exception e) {

e.printStackTrace();

displayErrorAlert("An error occurred. Please try again.");

}

}

private void handleWithdrawal() {

Stage withdrawalStage = new Stage();

withdrawalStage.setTitle("Withdrawal");

GridPane withdrawalGrid = new GridPane();

withdrawalGrid.setAlignment(Pos.CENTER);

withdrawalGrid.setHgap(10);

withdrawalGrid.setVgap(10);

withdrawalGrid.setPadding(new Insets(25, 25, 25, 25));

Text sceneTitle = new Text("Withdrawal");

sceneTitle.setFont(Font.font("Tahoma", 20));

withdrawalGrid.add(sceneTitle, 0, 0, 2, 1);

Label pin = new Label("Pin");

PasswordField pinField = new PasswordField();

withdrawalGrid.add(pin, 0, 1);

withdrawalGrid.add(pinField, 1, 1);

pinField.setPromptText("Pin Number");

Label wl2 = new Label("Amount:");

withdrawalGrid.add(wl2, 0, 2);

TextField amountField = new TextField();

amountField.setPromptText("Withdrawal Amount");

withdrawalGrid.add(amountField, 1, 2);

Button withdrawButton = new Button("Withdraw");

Button CancelButton = new Button("Cancel");

CancelButton.setStyle("-fx-background-color:slateblue;-fx-text-fill:white");

CancelButton.setCursor(Cursor.HAND);

HBox hbBtn = new HBox(10);

withdrawButton.setStyle("-fx-background-color:slateblue;-fx-text

-fill:white");

withdrawButton.setCursor(Cursor.HAND);

hbBtn.setAlignment(Pos.BOTTOM\_RIGHT);

hbBtn.getChildren().addAll(withdrawButton,CancelButton);

withdrawalGrid.add(hbBtn, 1, 3);

EventHandler<ActionEvent> can=new EventHandler<ActionEvent>(){

@Override

public void handle(ActionEvent t) {

withdrawalStage.close();

}

};

CancelButton.addEventFilter(ActionEvent.ACTION, can);

withdrawButton.setOnAction(e -> handleWithdrawAction(withdrawalStage,pinField, amountField));

withdrawalGrid.setStyle( "-fx-background-color: white; -fx-border-color: #cccccc; -fx-border-width: 1; -fx-border-radius: 5; -fx-padding: 20;");

Scene withdrawalScene = new Scene(withdrawalGrid, 400, 300); withdrawalStage.setScene(withdrawalScene); withdrawalStage.initModality(Modality.APPLICATION\_MODAL); withdrawalStage.showAndWait();

}

private void handleWithdrawAction(Stage withdrawalStage, PasswordField pinField, TextField amountField) {

try {

String pinNo = pinField.getText();

String amountStr = amountField.getText();

if (pinNo.isEmpty() || amountStr.isEmpty()) {

displayErrorAlert("Please fill in all fields.");

return;

}

int ac = user;

int pn = Integer.parseInt(pinNo);

int amt = Integer.parseInt(amountStr);

Connection con = DriverManager.getConnection

("jdbc:mysql://localhost:3306/login", "root", "r@m4112004");

String checkIfExistsQuery = "SELECT bal FROM logindetails WHERE accountNum= ? AND pin= ?";

try (PreparedStatement checkIfExistsStmt = con.prepareStatement

(checkIfExistsQuery)) {

checkIfExistsStmt.setInt(1, ac);

checkIfExistsStmt.setInt(2, pn);

ResultSet rs = checkIfExistsStmt.executeQuery();

if (rs.next()) {

int balance = rs.getInt("bal");

if(balance>=amt) {

// Update balance and insert transaction

String updateQuery = "UPDATE logindetails SET bal = bal - ? WHERE accountNum= ?";

try (PreparedStatement updateStatement = con.prepareStatement(updateQuery)) {

updateStatement.setInt(1, amt);

updateStatement.setInt(2, ac);

int rowsAffected = updateStatement.executeUpdate();

if (rowsAffected > 0)

{

balance -= amt;

Alert alert = new Alert(AlertType.INFORMATION);

alert.setTitle("Withdraw");

alert.setHeaderText(null);

alert.setContentText("Congratulations! Withdraw successful.Your Balance

is "+balance);

alert.showAndWait();

withdrawalStage.close();

}

else {

displayErrorAlert("Your balance is insufficient for withdrawal.");

}

}

String insertQuery = "INSERT INTO transaction(accountNum, pin,Toaccount, transactiontype, amount,timestamp) VALUES (?, ?, ?, ?,?,?)";

try (PreparedStatement insertStmt = con.prepareStatement(insertQuery))

{

insertStmt.setInt(1, ac);

insertStmt.setInt(2, pn);

insertStmt.setString(3,"Self");

insertStmt.setString(4, "Withdraw");

insertStmt.setInt(5, amt);

Timestamp currentTimestamp = new Timestamp(System.currentTimeMillis());

insertStmt.setTimestamp(6, currentTimestamp);

int rowsAffected = insertStmt.executeUpdate();

}

}

else

{

displayErrorAlert("Please check your balance and withdrawal amount");

pinField.setText("");

amountField.setText("");}

}

}

else {

displayErrorAlert("Please check your credentials.");

pinField.setText("");

amountField.setText("");

}

}

} catch (Exception e) {

e.printStackTrace();

displayErrorAlert("An error occurred. Please try again.");

}

}

private void displayErrorAlert(String message) {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle("Error");

alert.setHeaderText(null);

alert.setContentText(message);

alert.showAndWait();

}

private void showCreateAccountDialog(Stage stage) {

GridPane g1=new GridPane();

GridPane loginGrid = createLoginForm(primaryStage);g1.setStyle("-fx-background-color: white; -fx-border-color: #cccccc; -fx-border-width: 1; -fx- border-radius: 5; -fx-padding: 20;");

g1.setAlignment(Pos.CENTER);

g1.setHgap(10);

g1.setVgap(10);

g1.setPadding(new Insets(25, 25, 25, 25));

Text scenetitle = new Text("Account Creation");

scenetitle.setFont(Font.font("Tahoma", 20));

g1.add(scenetitle, 0, 0, 2, 1);

Label username=new Label("Username:");

TextField userText=new TextField();

Label password =new Label("Password");

Label email=new Label("Email");

TextField emailField=new TextField();

Label add=new Label("Address");

TextField addField=new TextField();

Label mobile=new Label("Mobile No.");

TextField mobileField=new TextField();

PasswordField passField=new PasswordField();

Label pin =new Label("Pin");

PasswordField pinField=new PasswordField();

Label amount=new Label("Initial Amount");

TextField amountField=new TextField();

Button createButton = new Button("Withdraw");

Button CancelButton = new Button("Cancel");

CancelButton.setStyle("-fx-background-color:slateblue;-fx-text-fill:white");

CancelButton.setCursor(Cursor.HAND);

HBox hbBtn = new HBox(10);

createButton.setStyle("-fx-background-color:slateblue;-fx-text-fill:white");

createButton.setCursor(Cursor.HAND);

hbBtn.setAlignment(Pos.BOTTOM\_RIGHT);

hbBtn.getChildren().addAll(createButton,CancelButton);

g1.add(username, 0, 2);

g1.add(userText, 1, 2);

g1.add(password,0,3);

g1.add(passField, 1, 3);

g1.add(email, 0, 4);

g1.add(emailField, 1, 4);

g1.add(add,0,5);

g1.add(addField,1,5);

g1.add(mobile,0,6);

g1.add(mobileField,1,6);

g1.add(pin, 0, 7);

g1.add(pinField, 1, 7);

g1.add(amount, 0, 8);

g1.add(amountField, 1, 8);

g1.add(hbBtn, 1, 9);

EventHandler<ActionEvent> can=new EventHandler<ActionEvent>(){

@Override

public void handle(ActionEvent t) {

stage.close();

BorderPane root = new BorderPane();

root.setStyle("-fx-background-color: white;");

GridPane loginGrid = createLoginForm(stage);

loginGrid.setStyle("-fx-background-color: white; -fx-border-color:#cccccc;

-fx-border-width: 1; -fx-border-radius: 5; -fx-padding: 20;");

root.setCenter(loginGrid);

// Create scene and set it on the stage

Scene scene = new Scene(root, 400, 300);

stage.setScene(scene);

// Show the stage

stage.show();

}

};

CancelButton.addEventFilter(ActionEvent.ACTION, can);

EventHandler<ActionEvent>e=new EventHandler<ActionEvent>()

{

@Override

public void handle(ActionEvent t) {

try {

String name = userText.getText();

String pass = passField.getText();

String mail = emailField.getText();

String pinNo = pinField.getText();

String amt=amountField.getText();

String ad=addField.getText();

String mob=mobileField.getText();

int pn = Integer.parseInt(pinNo);

int bal = Integer.parseInt(amt);

Connection con = DriverManager.getConnection

("jdbc:mysql://localhost:3306/login", "root", "r@m4112004");

String checkIfExistsQuery = "SELECT \* FROM logindetails WHERE email= ?";

try (PreparedStatement checkIfExistsStmt = con.prepareStatement

(checkIfExistsQuery))

{

checkIfExistsStmt.setString(1, mail);

ResultSet rs = checkIfExistsStmt.executeQuery();

if (rs.next())

{

Alert alert = new Alert(AlertType.ERROR);

alert.setTitle("Account creation Failed");

alert.setHeaderText(null);

alert.setContentText("This email is already registered. Please use

a different email.");

alert.showAndWait();

}

else

{

// Email doesn't exist, proceed with the insertion

user=accNo;

String insertQuery = "INSERT INTO logindetails(name,

email,accountNum,pin,password,bal,mobile,address,timestamp)

VALUES (?, ?, ?, ?,?,?,?,?,?)";

try (PreparedStatement insertStmt = con.prepareStatement

(insertQuery)) {

insertStmt.setString(1,name );

insertStmt.setString(2, mail);

insertStmt.setInt(3, accNo);

insertStmt.setInt(4, pn);

insertStmt.setString(5, pass);

insertStmt.setInt(6, bal);

insertStmt.setString(7,mob);

insertStmt.setString(8,ad);

Timestamp currentTimestamp = new Timestamp

(System.currentTimeMillis());

insertStmt.setTimestamp(9, currentTimestamp);

// Execute the INSERT query

int rowsAffected = insertStmt.executeUpdate();

if (rowsAffected > 0) {

Alert alert = new Alert(AlertType.INFORMATION);

alert.setTitle("Registration Successful");

alert.setHeaderText(null);

alert.setContentText("Congratulations! Account

created successfulLy.Your account number is "+accNo);

accNo++;

storeAccNoToFile();

alert.showAndWait();

stage.close();

showMainDashboard();

} else {

displayErrorAlert("Failed to insert the record.");

}

}

}

}

} catch (Exception e1) {

Alert alert = new Alert(AlertType.ERROR);

alert.setTitle("Error");

alert.setHeaderText(null);

alert.setContentText(e1.getMessage());

alert.showAndWait();

return;

}

}

};

createButton.addEventFilter(ActionEvent.ACTION, e);

Scene s=new Scene(g1,400,400);

stage.setScene(s);

stage.show();

}

private void storeAccNoToFile() {

try (BufferedWriter writer = new BufferedWriter(new FileWriter

("C:\\Users\\harin\\OneDrive\\Desktop\\java\\accNo.txt"))) {

writer.write(String.valueOf(accNo));

} catch (IOException e) {

e.printStackTrace();

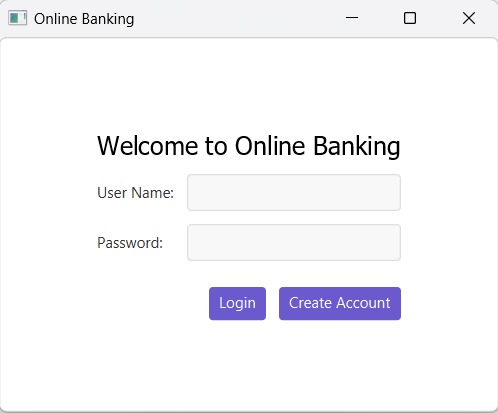
}

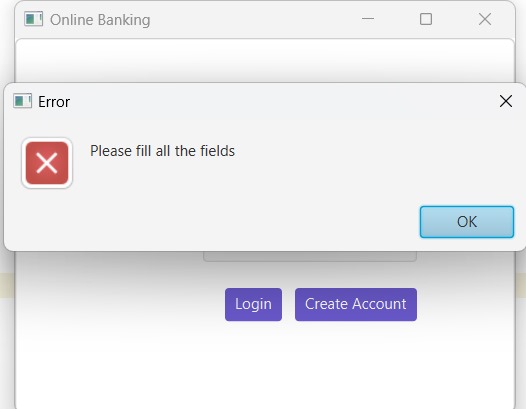
}

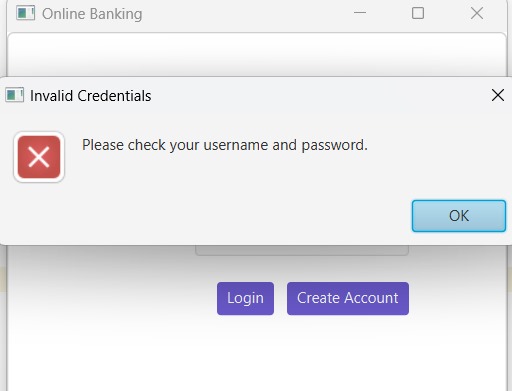
}

**6.OUTPUT**

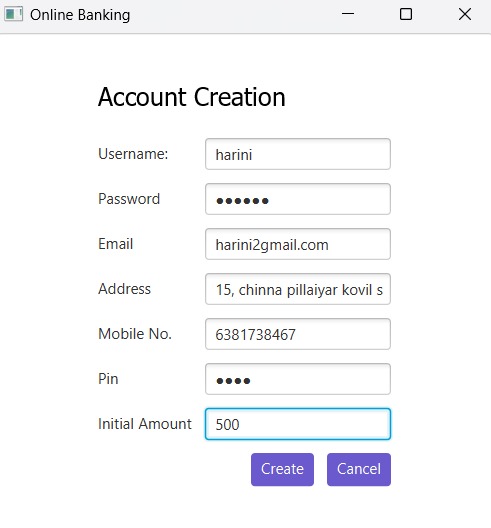
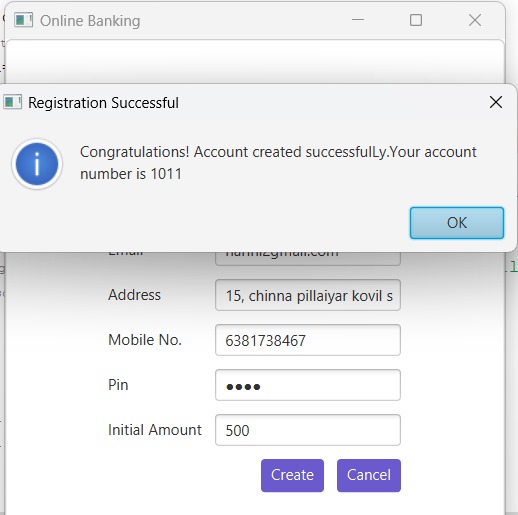
**Login page**



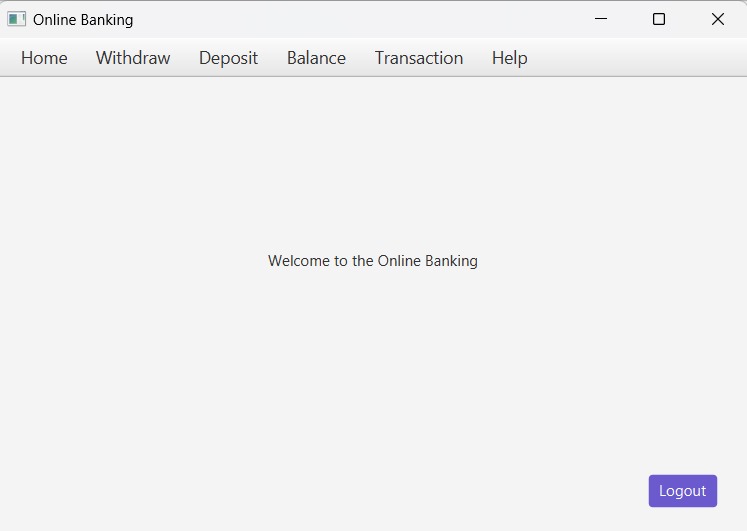




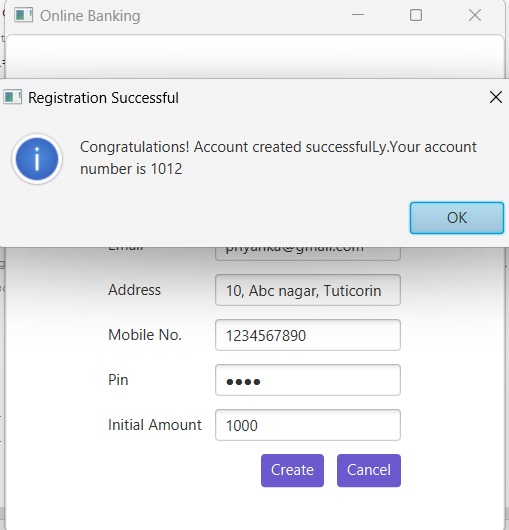
**Account creation**

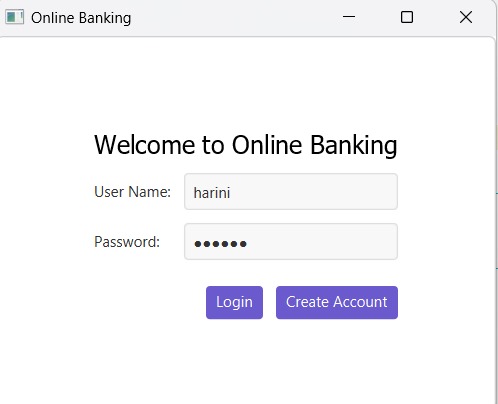
**After Account creation:**

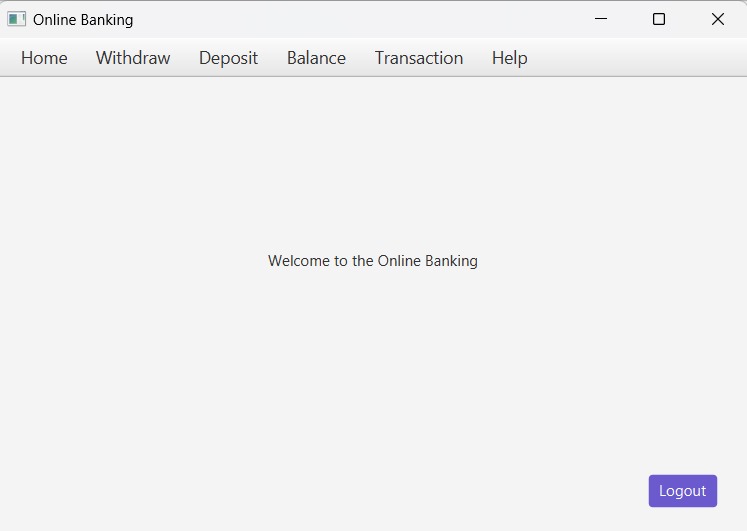


**Creating another new account**

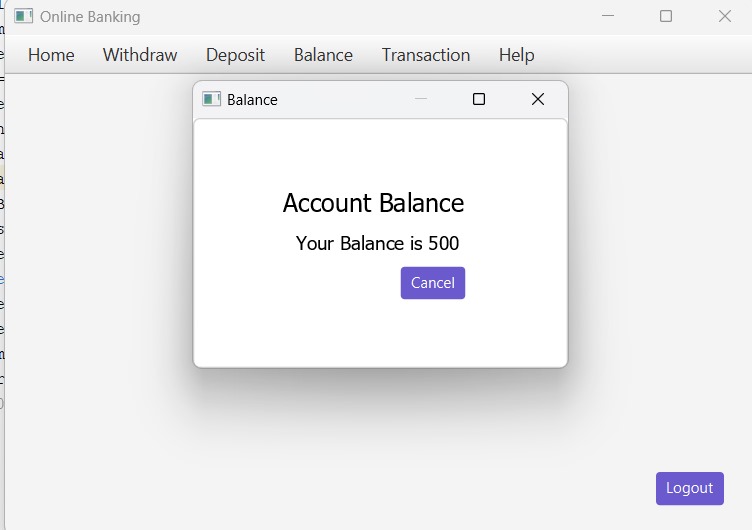


**Logging in again**

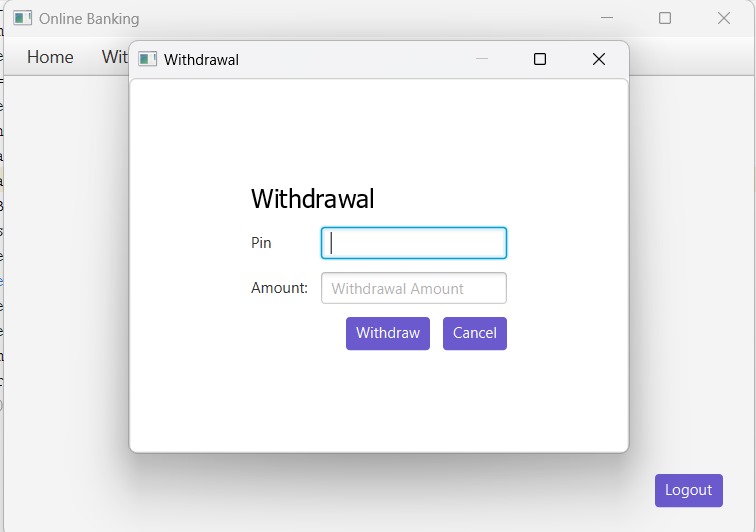




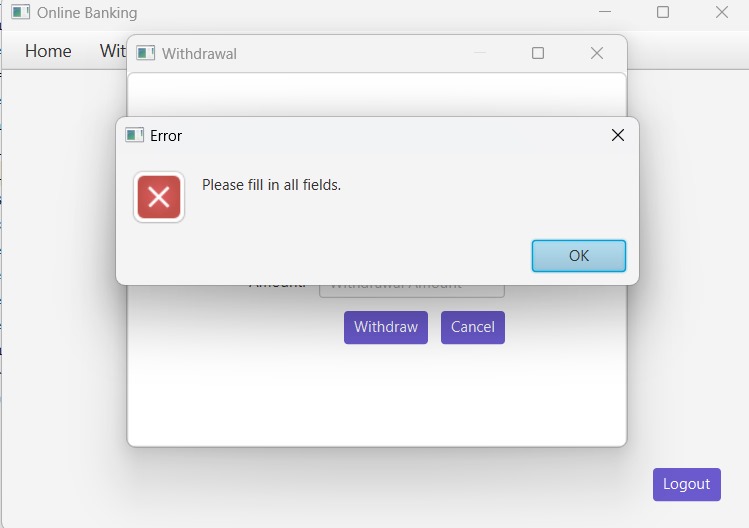
**Balance before withdraw**

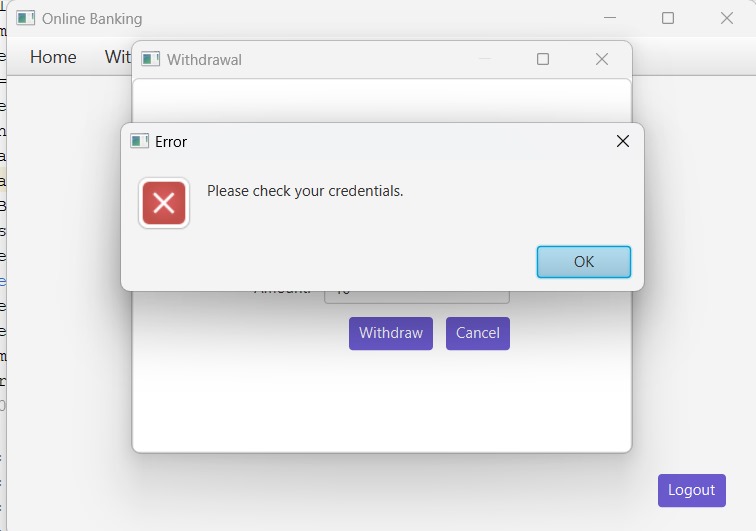


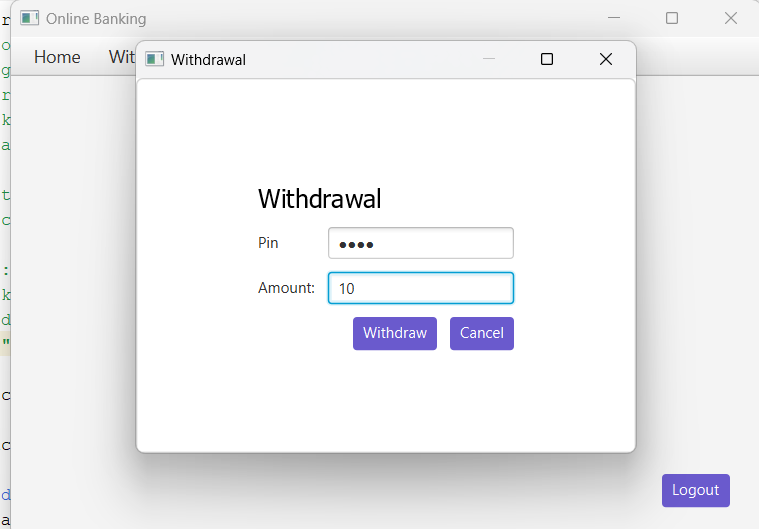
**Withdraw Option**

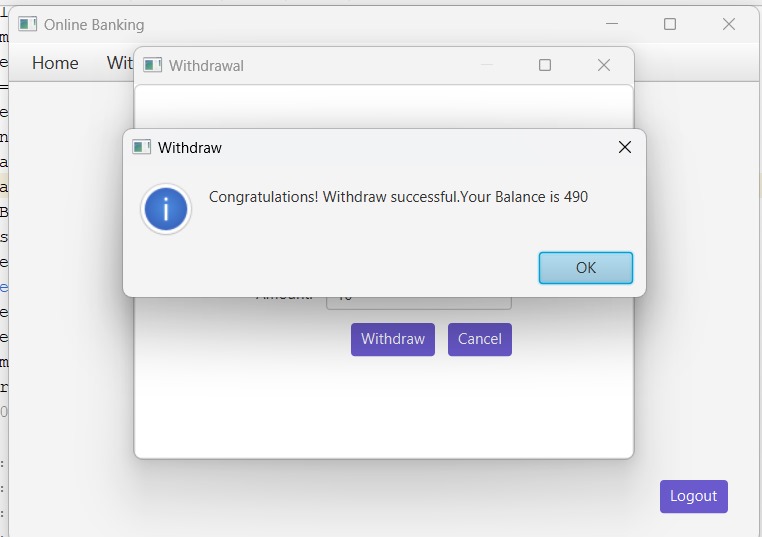


**If there is any empty field or any invalid credentials**

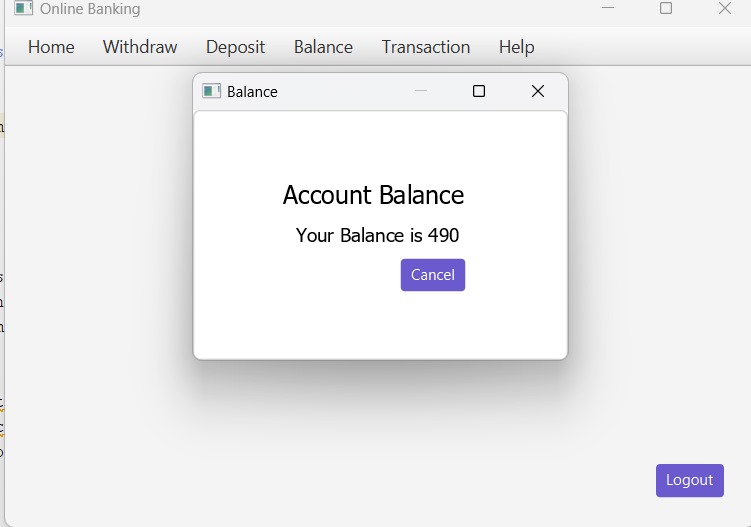




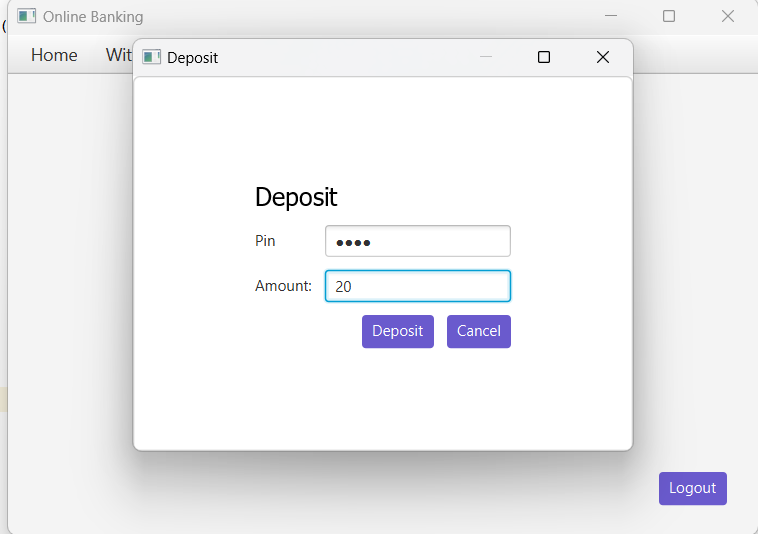
****

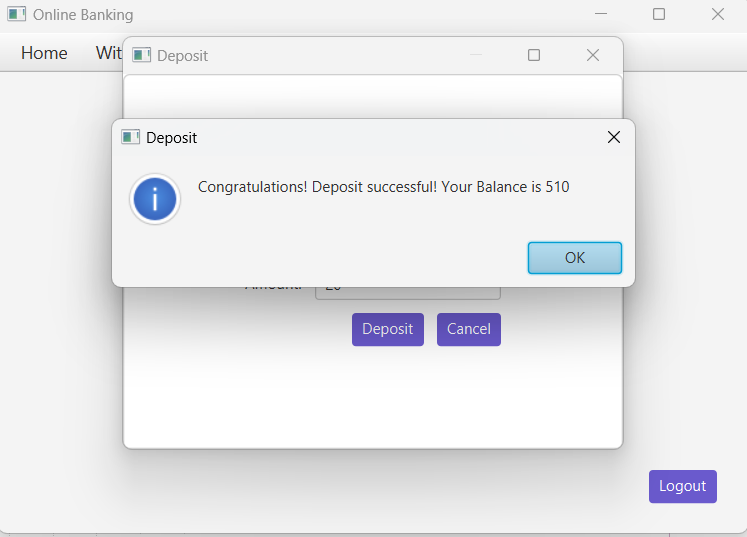


**Balance after withdraw**

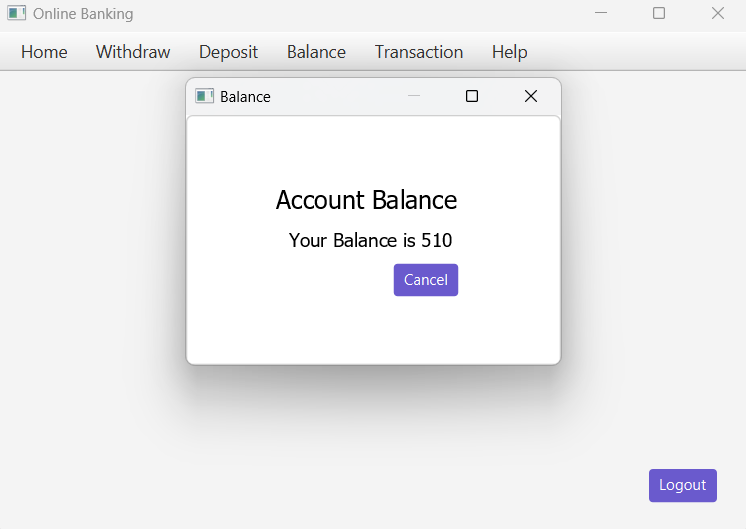


**Deposit option**

****

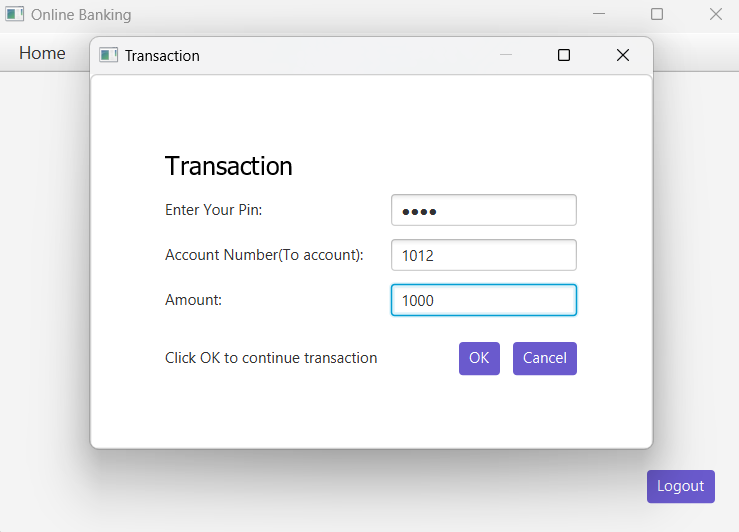
****

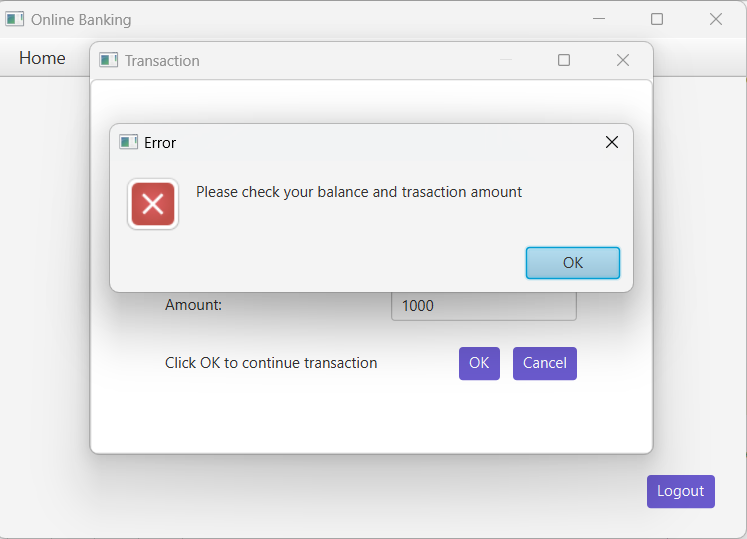
**Balance after deposit**

****

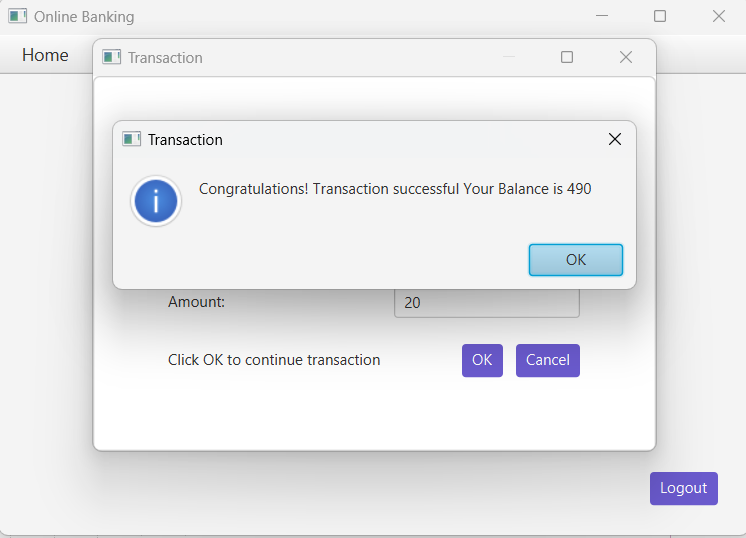
**Transaction**

**If balance is lesser than entered amount**

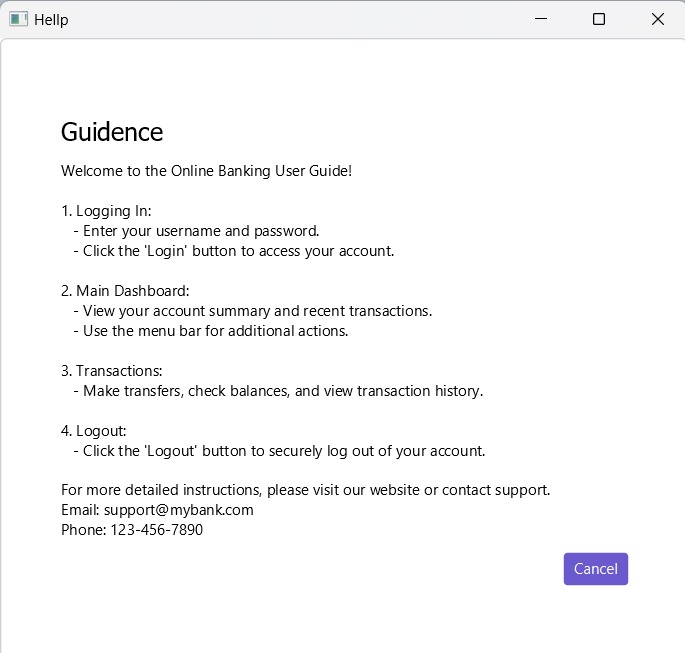
****

****

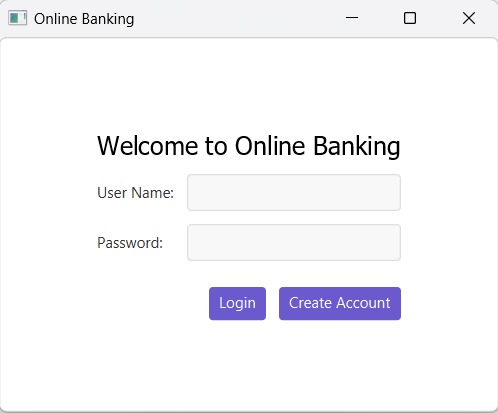
**Else**

****

**Help option**



**After logging out**



**7.CONCLUSION**

The Online Banking System represents not just a technological endeavor but a paradigm shift in modern banking practices. The seamless integration of Java and MySQL technologies provides a resilient, scalable, and user-centric platform, poised to redefine the banking experience. It addresses the dynamic requirements of contemporary banking, offering users a comprehensive and secure solution for managing their financial landscape.